

A Fundamental Tradeoff between Computation and Communication in Distributed Computing

Songze Li, *Student Member, IEEE*, Mohammad Ali Maddah-Ali, *Member, IEEE*, Qian Yu,
and A. Salman Avestimehr, *Member, IEEE*

Abstract

How can we optimally trade extra computing power to reduce the communication load in distributed computing? We answer this question by characterizing a fundamental tradeoff relationship between computation and communication in distributed computing, i.e., the two are *inverse-linearly* proportional to each other.

More specifically, a general distributed computing framework, motivated by commonly used structures like MapReduce, is considered, where the goal is to compute Q arbitrary output functions from N input files, by decomposing the overall computation into computing a set of “Map” and “Reduce” functions distributedly across K computing nodes. A coded scheme, named “Coded Distributed Computing” (CDC), is proposed to demonstrate that increasing the computation load of the Map phase by a factor of r (i.e., evaluating each Map function at r *carefully chosen* nodes) can create novel coding opportunities in the data shuffling phase that reduce the communication load by the same factor.

An information-theoretic lower bound on the communication load is also provided, which matches the communication load achieved by the CDC scheme. As a result, the optimal computation-communication tradeoff in distributed computing is exactly characterized.

Index Terms

Distributed Computing, MapReduce, Computation-Communication Tradeoff, Coded Multicasting

I. INTRODUCTION

We consider a general distributed computing framework, motivated by prevalent structures like MapReduce [3] and Spark [4], in which the overall computation is decomposed into two stages: “Map” and “Reduce”. Firstly in the Map stage, distributed computing nodes process parts of the input data locally, generating some intermediate values according to their designed Map functions. Next, they exchange the calculated intermediate values among

S. Li, Q. Yu and A.S. Avestimehr are with the Department of Electrical Engineering, University of Southern California, Los Angeles, CA, 90089, USA (e-mail: songzeli@usc.edu; qyu880@usc.edu; avestimehr@ee.usc.edu).

M. A. Maddah-Ali is with Bell Laboratories, Nokia, Holmdel, NJ, 07733, USA (e-mail: mohammad.maddah-ali@nokia.com).

A preliminary part of this work was presented in 53rd Annual Allerton Conference on Communication, Control, and Computing, 2015 [1]. A part of this work will be presented in IEEE International Symposium on Information Theory, 2016 [2].

each other (a.k.a. data shuffling), in order to calculate the final output results distributedly using their designed Reduce functions.

Within this framework, data shuffling often appears to limit the performance of distributed computing applications, including *tera-sort* [5], *ranked-inverted-index* [6] and machine learning algorithms [7]. In a Facebook’s Hadoop cluster, it is observed that 33% of the overall job execution time is spent on data shuffling [7]. As such motivated, we ask this fundamental question that *if coding can help distributed computing in reducing the load of communication and speeding up the overall computation?* Coding is known to be helpful in coping with the channel uncertainty in telecommunication and also in reducing the storage cost in distributed storage systems and cache networks. In this work, we extend the application of coding to *distributed computing* and propose a framework to substantially reduce the load of data shuffling via coding and some extra computing in the Map phase.

More specifically, we formulate and characterize a fundamental tradeoff relationship between “computation load” in the Map phase and “communication load” in the data shuffling phase, and demonstrate that the two are *inverse-linearly* proportional to each other. We propose an optimal coded scheme, named “Coded Distributed Computing” (CDC), which demonstrates that increasing the computation load of the Map phase by a factor of r (i.e., evaluating each Map function at r *carefully chosen* nodes) can create novel coding opportunities in the data shuffling phase that reduce the communication load by the same factor.

To illustrate our main result, consider a distributed computing framework to compute Q arbitrary output functions from N input files, using K distributed computing nodes. As mentioned earlier, the overall computation is performed by computing a set of Map and Reduce functions distributedly across the K nodes. In the Map phase, each input file is processed locally, in one of the nodes, to generate Q intermediate values, each corresponding to one of the Q output functions. Thus, at the end of this phase, QN intermediate values are calculated, which can be split into Q subsets of N intermediate values and each subset is needed to calculate one of the output functions. In the Shuffle phase, for every output function to be calculated, all N intermediate values corresponding to that function are transferred to one of the nodes for reduction. Of course, depending on the node that has been chosen to reduce an output function, a part of the intermediate values are already available locally, and do not need to be transferred in the Shuffle phase. This is because that the Map phase has been carried out on the same set of nodes, and the results of mapping done at a node can remain in that node to be used for the Reduce phase. This offers some saving in the load of communication. To reduce the communication load even more, we may map each input file in *more than one* nodes. Apparently, this increases the fraction of intermediate values that are locally available. However, as we will show, there is a better way to exploit this redundancy in computation to reduce the communication load. The main message of this paper is to show that following a particular pattern in repeating Map computations along with some coding techniques, we can significantly reduce the load of communication. Perhaps surprisingly, we show that the gain of coding in reducing communication load scales with the size of the network.

To be more precise, we define the *computation load* r , $1 \leq r \leq K$, as the total number of computed Map functions at the nodes, normalized by N . For example, $r = 1$ means that none of the Map functions has been re-computed, and $r = 2$ means that on average each Map function can be computed on two nodes. We also define

communication load L , $0 \leq L \leq 1$, as the total amount of information exchanged across nodes in the shuffling phase, normalized by the size of QN intermediate values, in order to compute the Q output functions disjointly and uniformly across the K nodes. Based on this formulation, we now ask the following fundamental question:

- Given a computation load r in the Map phase, what is the minimum communication load $L^*(r)$, using any data shuffling scheme, needed to compute the final output functions?

We propose Coded Distributed Computing (CDC) that achieves a communication load of $L_{\text{coded}}(r) = \frac{1}{r} \cdot (1 - \frac{r}{K})$ for $r = 1, \dots, K$, and the lower convex envelop of these points. CDC employs a specific strategy to assign the computations of the Map and Reduce functions across the computing nodes, in order to enable novel coding opportunities for data shuffling. In particular, for a computation load $r \in \{1, \dots, K\}$, CDC utilizes a carefully designed repetitive mapping of data blocks at r distinct nodes to create coded multicast messages that deliver data *simultaneously* to a subset of $r \geq 1$ nodes. Hence, compared with an uncoded data shuffling scheme, which as we show later achieves a communication load $L_{\text{uncoded}}(r) = 1 - \frac{r}{K}$, CDC is able to reduce the communication load by exactly a factor of the computation load r . Furthermore, the proposed CDC scheme applies to a more general distributed computing framework where every output function is computed by more than one, or particularly $s \in \{1, \dots, K\}$ nodes, which provides better fault-tolerance in distributed computing.

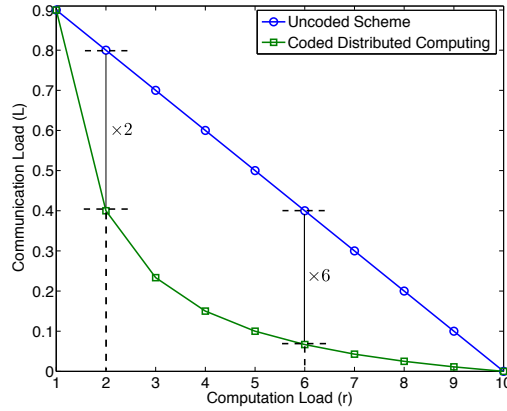


Fig. 1: Comparison of the communication load achieved by Coded Distributed Computing $L_{\text{coded}}(r)$ with that of the uncoded scheme $L_{\text{uncoded}}(r)$, for $Q = 10$ output functions, $N = 2520$ input files and $K = 10$ computing nodes. For $r \in \{1, \dots, K\}$, CDC is r times better than the uncoded scheme.

We numerically compare the computation-communication tradeoffs of CDC and uncoded data shuffling schemes (i.e., $L_{\text{coded}}(r)$ and $L_{\text{uncoded}}(r)$) in Fig. 1. As it is illustrated, in the uncoded scheme that achieves a communication load $L_{\text{uncoded}}(r) = 1 - \frac{r}{K}$, increasing the computation load r offers only a modest reduction in communication load. In fact for any r , this gain vanishes for large number of nodes K . Consequently, it is not justified to trade computation for communication using uncoded schemes. However, for the coded scheme that achieves a communication load of $L_{\text{coded}}(r) = \frac{1}{r} \cdot (1 - \frac{r}{K})$, increasing the computation load r will significantly reduce the communication load, and this gain does not vanish for large K . For example as illustrated in Fig. 1, when mapping each file at one extra node ($r = 2$), CDC reduces the communication load by 55.6%, while the uncoded scheme only reduces it by

11.1%.

We also prove an information-theoretic lower bound on the minimum communication load $L^*(r)$. To prove the lower bound, we derive a lower bound on the total number of bits communicated by any subset of nodes, using induction on the size of the subset. To derive the lower bound for a particular subset of nodes, we first establish a lower bound on the number of bits needed by one of the nodes to recover the intermediate values it needs to calculate its assigned output functions, and then utilize the bound on the number of bits communicated by the rest of the nodes in that subset, which is given by the inductive argument. The derived lower bound on $L^*(r)$ matches the communication load achieved by the CDC scheme for any computation load $1 \leq r \leq K$. As a result, we *exactly* characterize the optimal tradeoff between computation load and communication load in the following:

$$L^*(r) = L_{\text{coded}}(r) = \frac{1}{r} \cdot \left(1 - \frac{r}{K}\right), r \in \{1, \dots, K\}.$$

For general $1 \leq r \leq K$, $L^*(r)$ is the lower convex envelop of the above points $\{(r, L_{\text{coded}}(r)) : r \in \{1, \dots, K\}\}$. Note that for large K , $\frac{1}{r} \cdot \left(1 - \frac{r}{K}\right) \approx \frac{1}{r}$, hence $L^*(r) \approx \frac{1}{r}$. This result reveals a fundamental linear-inverse proportionality relationship between computation load and communication load in distributed computing. This also illustrates that the gain of $\frac{1}{r}$ achieved by CDC is optimal and it cannot be improved by any other scheme (since $L_{\text{coded}}(r)$ is an information-theoretic lower bound on $L^*(r)$ that applies to any data shuffling scheme).

Finally, we discuss and address some of the practical challenges in implementing CDC in current systems. In particular, we consider the multicasting capability of the network, the flexibility of data storage for computation, and parallelizing the Map phase and the Shuffle phase of the considered framework.

Related Works. The problem of characterizing the minimum communication for distributed computing has been previously considered in several settings in both computer science and information theory communities. In [8], a basic computing model is proposed, where two parties have x and y and aim to compute a boolean function $f(x, y)$ by exchanging the minimum number of bits between them. Also, the problem of minimizing the required communication for computing the modulo-two sum of distributed binary sources with symmetric joint distribution was introduced in [9]. Following these two seminal works, a wide range of communication problems in the scope of distributed computing have been studied (cf. [10]–[15]). The key differences distinguishing the setting in this paper from most of the prior ones are 1) We focus on the flow of communication in a general distributed computing framework, motivated by MapReduce, rather than the structures of the functions or the input distributions. 2) We do not impose any constraint on the numbers of output results, input data files and computing nodes (they can be arbitrarily large), 3) We do not assume any special property (e.g. linearity) of the computed functions.

The idea of efficiently creating and exploiting *coded multicasting* was initially proposed in the context of cache networks in [16], [17], and extended in [18], [19], where caches pre-fetch part of the content in a way to enable coding during the content delivery, minimizing the network traffic. In this paper, we propose a framework to study the tradeoff between computation and communication in distributed computing. We demonstrate that the coded multicasting opportunities exploited in the above caching problems also exist in the data shuffling of distributed computing frameworks, which can be created by a strategy of repeating the computations of the Map functions

specified by the Coded Distributed Computing (CDC) scheme.

Finally, in a recent work [20], the authors have proposed methods for utilizing codes to speed up some specific distributed machine learning algorithms. The considered problem in this paper differs from [20] in the following aspects. We propose a general methodology for utilizing coding in data shuffling that can be applied to any distributed computing framework with a MapReduce structure, regardless of the underlying application. In other words, any distributed computing algorithm that fits in the MapReduce framework can benefit from the proposed CDC solution. We also characterize the information-theoretic computation-communication tradeoff in such frameworks. Furthermore, the coding used in [20] is at the application layer (i.e., applying computation on coded data), while in this paper we focus on applying codes directly on the shuffled data.

II. PROBLEM FORMULATION

In this section, we formulate a general distributed computing framework motivated by MapReduce, and define the function characterizing the tradeoff between computation and communication.

For some system parameters $Q, N, K \in \mathbb{N}$, we consider the problem of computing Q arbitrary output functions from N input files using a cluster of K distributed computing nodes (servers). More specifically, given N input files $w_1, \dots, w_N \in \mathbb{F}_{2^F}$, for some $F \in \mathbb{N}$, the goal is to compute Q output functions ϕ_1, \dots, ϕ_Q , where $\phi_q : (\mathbb{F}_{2^F})^N \rightarrow \mathbb{F}_{2^B}$, $q \in \{1, \dots, Q\}$ maps all input files to a length- B binary stream $u_q = \phi_q(w_1, \dots, w_N) \in \mathbb{F}_{2^B}$, for some $B \in \mathbb{N}$.

Motivated by MapReduce, we assume that as illustrated in Fig. 2 the computation of the output function ϕ_q , $q \in \{1, \dots, Q\}$ can be decomposed as follows:

$$\phi_q(w_1, \dots, w_N) = h_q(g_{q,1}(w_1), \dots, g_{q,N}(w_N)), \quad (1)$$

where

- The “Map” functions $\vec{g}_n = (g_{1,n}, \dots, g_{Q,n}) : \mathbb{F}_{2^F} \rightarrow (\mathbb{F}_{2^T})^Q$, $n \in \{1, \dots, N\}$ maps the input file w_n into Q length- T intermediate values $v_{q,n} = g_{q,n}(w_n) \in \mathbb{F}_{2^T}$, $q \in \{1, \dots, Q\}$, for some $T \in \mathbb{N}$.
- The “Reduce” functions $h_q : (\mathbb{F}_{2^T})^N \rightarrow \mathbb{F}_{2^B}$, $q \in \{1, \dots, Q\}$ maps the intermediate values of the output function ϕ_q in all input files into the output value $u_q = h_q(v_{q,1}, \dots, v_{q,N})$.

Remark 1. Note that for every set of output functions ϕ_1, \dots, ϕ_Q such a Map-Reduce decomposition exists (e.g., setting $g_{q,n}$'s to identity and h_q to ϕ_q in (1)). However, such a decomposition is not unique, and in the distributed computing literature, there has been quite some work on developing appropriate decompositions of computations like join, sorting and matrix multiplication (cf. [3], [21]), for them to be performed efficiently in a distributed manner. Here we do not impose any constraint on how the Map and Reduce functions are chosen (for example, they can be arbitrary linear or non-linear functions). \square

The above computation is carried out by K distributed computing nodes, labelled as Node 1, \dots , Node K . They are interconnected through a multicast network. Following the above decomposition, the computation proceeds in three phases: *Map*, *Shuffle* and *Reduce*.

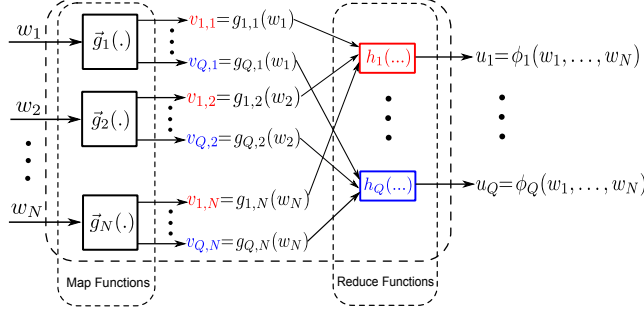


Fig. 2: Illustration of a two-stage distributed computing framework. The overall computation is decomposed into computing a set of Map and Reduce functions.

Map Phase: Node k , $k \in \{1, \dots, K\}$ computes the Map functions of a set of files \mathcal{M}_k , which are stored on Node k , for some design parameter $\mathcal{M}_k \subseteq \{w_1, \dots, w_N\}$. For each file w_n in \mathcal{M}_k , Node k computes $\vec{g}_n(w_n) = (v_{1,n}, \dots, v_{Q,n})$. We assume that each file is mapped by at least one node, i.e., $\bigcup_{k=1, \dots, K} \mathcal{M}_k = \{w_1, \dots, w_N\}$.

Definition 1 (Computation Load). We define the *computation load*, denoted by r , $1 \leq r \leq K$, as the total number of Map functions computed across the K nodes, normalized by the number of files N , i.e., $r \triangleq \frac{\sum_{k=1}^K |\mathcal{M}_k|}{N}$. The computation load r can be interpreted as the average number of nodes that map each file. \diamond

Shuffle Phase: Node k , $k \in \{1, \dots, K\}$ is responsible for computing a subset of output functions, whose indices are denoted by a set $\mathcal{W}_k \subseteq \{1, \dots, Q\}$. The computations of the output functions are assigned *uniformly* and *disjointly* across the K nodes, such that 1) $|\mathcal{W}_1| = \dots = |\mathcal{W}_K| = \frac{Q}{K} \in \mathbb{N}$, 2) $\mathcal{W}_j \cap \mathcal{W}_k = \emptyset$ for $j \neq k$. To compute the output value u_q for some $q \in \mathcal{W}_k$, Node k needs the intermediate values that are *not* computed *locally* in the Map phase, i.e., $\{v_{q,n} : q \in \mathcal{W}_k, w_n \notin \mathcal{M}_k\}$. After Node k , $k \in \{1, \dots, K\}$ has finished mapping all the files in \mathcal{M}_k , the K nodes proceed to exchange the needed intermediate values. We formally define a *shuffling scheme* as follows:

- Each node k , $k \in \{1, \dots, K\}$, creates a message X_k as a function of the intermediate values computed locally during the Map phase, i.e., $X_k = \psi_k(\{\vec{g}_n : w_n \in \mathcal{M}_k\})$, and multicasts it to a subset of $1 \leq j \leq K-1$ nodes.

Definition 2 (Communication Load). We define the *communication load*, denoted by L , $0 \leq L \leq 1$, as the number of bits communicated by the K nodes during the Shuffle phase, normalized by QNT , which is the total number of bits in all intermediate values $\{v_{q,n} : q \in \{1, \dots, Q\}, n \in \{1, \dots, N\}\}$. \diamond

Reduce Phase: Node k , $k \in \{1, \dots, K\}$ uses the local results from the Map phase $\{\vec{g}_n : w_n \in \mathcal{M}_k\}$ and the received messages X_1, \dots, X_K in the Shuffle phase to construct the inputs to the corresponding Reduce functions of \mathcal{W}_k , and calculates $u_q = h_q(v_{q,1} \dots v_{q,N})$ for all $q \in \mathcal{W}_k$.

We say that a computation-communication pair $(r, L) \in \mathbb{R} \times \mathbb{R}$ is *feasible* if there exist $\mathcal{M}_1, \dots, \mathcal{M}_K$, $\mathcal{W}_1, \dots, \mathcal{W}_K$ and a shuffling scheme such that Node k can successfully compute all the output functions whose indices are in \mathcal{W}_k , for all $k \in \{1, \dots, K\}$.

Definition 3. We define the *computation-communication function* of the distributed computing framework

$$L^*(r) \triangleq \inf\{L : (r, L) \text{ is feasible}\}. \quad (2)$$

$L^*(r)$ characterizes the optimal tradeoff between computation and communication in this framework. \diamond

Example (Uncoded Scheme). In the Shuffle phase of a simple “uncoded” scheme, each node receives the needed intermediate values sent uncodedly by some other nodes. Since a total of QN intermediate values are needed across the K nodes and $rN \cdot \frac{Q}{K} = \frac{rQN}{K}$ of them are already available after the Map phase, the communication load achieved by the uncoded scheme

$$L_{\text{uncoded}}(r) = 1 - r/K. \quad (3)$$

Remark 2. After the Map phase, each node knows the intermediate values of *all* Q output functions in the files it has mapped. Therefore, the data requirements of all valid assignments of the Reduce functions, specified by $\mathcal{W}_1, \dots, \mathcal{W}_K$, can be satisfied with *identical* communication loads (the shuffling schemes are different though). In other words, the communication load is independent of the assignment of the Reduce functions. \square

In this paper, we also consider a generalization of the above framework, which we call “cascaded distributed computing framework”, where after the Map phase, each Reduce function is computed by more than one or particularly s nodes, for some $s \in \{1, \dots, K\}$. This generalized model is motivated by the fact that many distributed computing jobs require multiple rounds of Map and Reduce computations, where the Reduce results of the previous round serve as the inputs to the Map functions of the next round. Computing each Reduce function at more than one node admits *data redundancy* for the subsequent Map-function computations, which can help to improve the fault-tolerance and reduce the communication load of the next-round data shuffling. In particular, we consider the case where the number of output functions Q is sufficiently large, and focus on a uniform distribution of the computations of the Reduce functions, such that every subset of s nodes compute a disjoint subset of $\frac{Q}{\binom{K}{s}} \in \mathbb{N}$ Reduce functions.

The feasible computation-communication triple $(r, s, L) \in \mathbb{R} \times \mathbb{N} \times \mathbb{R}$ is defined similar as before. We define the computation-communication function of the cascaded distributed computing framework

$$L^*(r, s) \triangleq \inf\{L : (r, s, L) \text{ is feasible}\}. \quad (4)$$

III. MAIN RESULTS

Theorem 1. The computation-communication function of the distributed computing framework, $L^*(r)$ is given by

$$L^*(r) = L_{\text{coded}}(r) \triangleq \frac{1}{r} \cdot \left(1 - \frac{r}{K}\right), \quad r \in \{1, \dots, K\}, \quad (5)$$

for sufficiently large N . For general $1 \leq r \leq K$, $L^*(r)$ is the lower convex envelop of the above points $\{(r, \frac{1}{r} \cdot (1 - \frac{r}{K})) : r \in \{1, \dots, K\}\}$.

To prove the achievability, we propose a *coded* scheme in the next section, namely Coded Distributed Computing, which achieves the communication load $L_{\text{coded}}(r) = \frac{1}{r} \cdot (1 - \frac{r}{K})$ for integer-valued computation load r . We

demonstrate that no other scheme can achieve a communication load smaller than the lower convex envelop of the points $\{(r, \frac{1}{r} \cdot (1 - \frac{r}{K})) : r \in \{1, \dots, K\}\}$ by proving the converse in Section V.

Remark 3. Theorem 1 exactly characterizes the optimal tradeoff between the computation load and the communication load in the considered distributed computing framework. \square

Remark 4. For $r \in \{1, \dots, K\}$, the communication load achieved in Theorem 1 is less than that of the uncoded scheme in (3) by a multiplicative factor of r , which equals the computation load and can grow unboundedly as the number of nodes K increases if e.g. $r = \Theta(K)$. As illustrated in Fig. 1 in Section I, while the communication load of the uncoded scheme decreases linearly as the computation load increases, $L_{\text{coded}}(r)$ achieved in Theorem 1 is inverse-linearly proportional to the computation load. \square

Remark 5. While increasing the computation load r causes a longer Map phase, the coded achievable scheme of Theorem 1 maximizes the reduction of the communication load using the extra computations. Therefore, Theorem 1 provides an analytical framework to optimally allocate the computation and communication resources, minimizing the overall job execution time. \square

Theorem 2. *The computation-communication function of the cascaded distributed computing framework, $L^*(r, s)$ is characterized by*

$$L^*(r, s) = L_{\text{coded}}(r, s) \triangleq \sum_{\ell=\max\{r+1, s\}}^{\min\{r+s, K\}} \frac{\ell(K) \binom{\ell-2}{r-1} \binom{r}{\ell-s}}{r \binom{K}{r} \binom{K}{s}}, \quad r \in \{1, \dots, K\}, \quad (6)$$

for sufficiently large Q and N , and $s \in \{1, \dots, K\}$. For general $1 \leq r \leq K$, $L^*(r, s)$ is the lower convex envelop of the above points $\{(r, L_{\text{coded}}(r, s)) : r \in \{1, \dots, K\}\}$.

Remark 6. A preliminary part of this result, in particular the achievability for the special case of $s = 1$, or the achievable scheme of Theorem 1 was presented in [1]. We note that when $s = 1$, Theorem 2 provides the same result as in Theorem 1, i.e., $L^*(r, 1) = \frac{1}{r} \cdot (1 - \frac{r}{K})$, for $r \in \{1, \dots, K\}$. \square

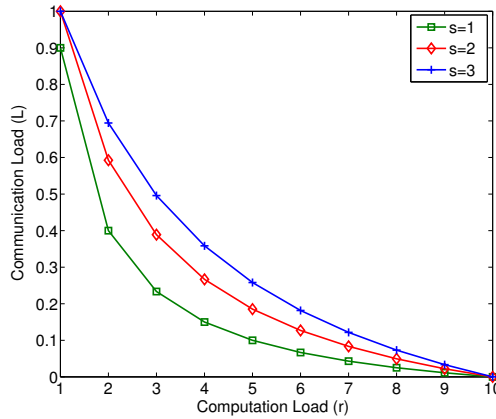


Fig. 3: Minimum communication load $L^*(r, s) = L_{\text{coded}}(r, s)$ in Theorem 2, for $Q=360$ output functions, $N=2520$ input files and $K=10$ computing nodes.

Remark 7. For any fixed $s \in \{1, \dots, K\}$ (number of nodes that compute each Reduce function), as illustrated in Fig. 3, the communication load achieved in Theorem 2 outperforms the linear relationship between computation and communication, i.e., it is superlinear with respect to the computation load r . \square

IV. ACHIEVABILITY: CODED DISTRIBUTED COMPUTING

In this section, we prove the upper bounds in Theorem 1 and 2 by presenting and analyzing a coded scheme, which we call Coded Distributed Computing (CDC). We focus on the more general case considered in Theorem 2 with $s \geq 1$, and the scheme for Theorem 1 simply follows by setting $s = 1$.

We first consider the integer-valued computation load $r \in \{1, \dots, K\}$, and then generalize the CDC scheme for any $1 \leq r \leq K$. When $r = K$, every node can map all the input files and compute all the output functions locally, thus no communication is needed and $L^*(K, s) = 0$ for all $s \in \{1, \dots, K\}$. In what follows, we focus on the case where $r < K$.

Along the presentation of the general CDC scheme, we use an illustrative example to demonstrate the key ingredients of the scheme.

Example (Coded Distributed Computing). We illustrate the key ideas of CDC using an example with $Q = 6$ output functions, $N = 6$ input files, and $K = 4$ nodes. We consider the case where the computation load $r = 2$, and each Reduce function is computed by $s = 2$ nodes. \square

A. Map Phase Design

We assume that the number of input files N is sufficiently large such that $N = \binom{K}{r} \eta_1$ for some $\eta_1 \in \mathbb{N}$.¹ In the Map phase the N input files are evenly partitioned into $\binom{K}{r}$ disjoint batches of size η_1 , each corresponding to a subset $\mathcal{T} \subset \{1, \dots, K\}$ of size r :

$$\{w_1, \dots, w_N\} = \{\mathcal{B}_{\mathcal{T}} : \mathcal{T} \subset \{1, \dots, K\}, |\mathcal{T}| = r\}, \quad (7)$$

where $\mathcal{B}_{\mathcal{T}}$ denotes the batch of η_1 files corresponding to the subset \mathcal{T} .

Given this partition, Node k , $k \in \{1, \dots, K\}$ computes the Map functions of the files in $\mathcal{B}_{\mathcal{T}}$ if $k \in \mathcal{T}$. Or equivalently, $\mathcal{B}_{\mathcal{T}} \subseteq \mathcal{M}_k$ if $k \in \mathcal{T}$. Since each node is in $\binom{K-1}{r-1}$ subsets of size r , each node computes $\binom{K-1}{r-1} \eta_1 = \frac{rN}{K}$ Map functions, i.e., $|\mathcal{M}_k| = \frac{rN}{K}$ for all $k \in \{1, \dots, K\}$. After the Map phase, Node k , $k \in \{1, \dots, K\}$ knows the intermediate values of all Q output functions in the files in \mathcal{M}_k , i.e., $\{v_{q,n} : q \in \{1, \dots, Q\}, w_n \in \mathcal{M}_k\}$.

Example (Coded Distributed Computing: Map Phase Design). In the example, the Map phase is designed such that every $r = 2$ nodes map a common file. More precisely as illustrated in Fig. 4, the sets of the files mapped by the 4 nodes are $\mathcal{M}_1 = \{w_1, w_2, w_3\}$, $\mathcal{M}_2 = \{w_1, w_4, w_5\}$, $\mathcal{M}_3 = \{w_2, w_4, w_6\}$, and $\mathcal{M}_4 = \{w_3, w_5, w_6\}$. After the Map phase, Node k , $k \in \{1, 2, 3, 4\}$ knows the intermediate values of all $Q = 6$ output functions in the files in \mathcal{M}_k , i.e., $\{v_{q,n} : q \in \{1, \dots, 6\}, w_n \in \mathcal{M}_k\}$. \square

¹For small number of files $N < \binom{K}{r}$, we can apply the CDC scheme to a subset of nodes with less computation load, achieving a part of the gain in reducing the communication load.

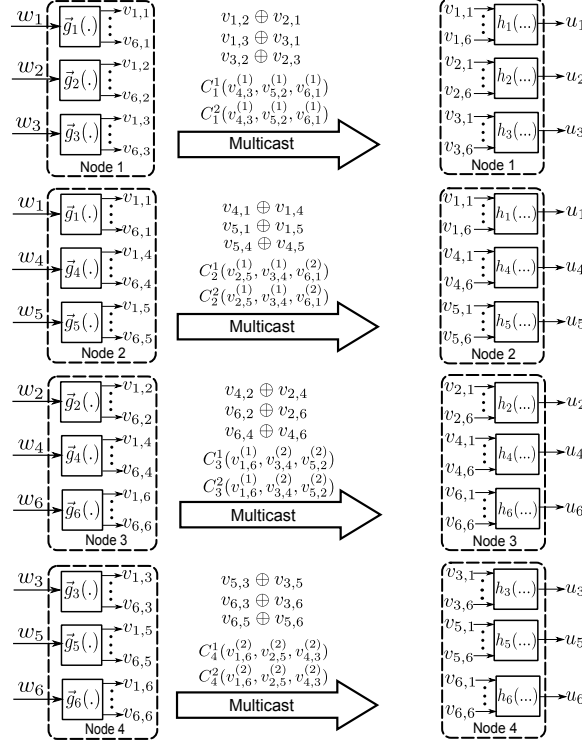


Fig. 4: Illustration of the CDC scheme to compute $Q = 6$ output functions from $N = 6$ input files distributedly at $K = 4$ computing nodes. Each file is mapped by $r = 2$ nodes and each output function is computed by $s = 2$ nodes. After the Map phase, every node knows 6 intermediate values, one for each output function, in every file it has mapped. In the Shuffle phase, Node k , $k \in \{1, 2, 3, 4\}$ splits the locally computed intermediate values $v_{q,n}$ evenly into 2 segments $v_{q,n} = (v_{q,n}^{(1)}, v_{q,n}^{(2)})$, and multicasts 2 random linear combinations of the segments $C_k^1(\cdot, \cdot, \cdot)$, $C_k^2(\cdot, \cdot, \cdot)$.

B. Coded Data Shuffling

We assume that the number of the output functions Q are sufficiently large such that $Q = \binom{K}{s} \eta_2$ for some $\eta_2 \in \mathbb{N}$. The computations of the Reduce functions are assigned uniformly across the K nodes as follows. Firstly the Q Reduce functions are evenly partitioned into $\binom{K}{s}$ disjoint batches of size η_2 , each corresponding to a unique subset \mathcal{P} of s nodes:

$$\{1, \dots, Q\} = \{\mathcal{D}_{\mathcal{P}} : \mathcal{P} \subseteq \{1, \dots, K\}, |\mathcal{P}| = s\}, \quad (8)$$

where $\mathcal{D}_{\mathcal{P}}$ denotes the indices of the batch of η_2 Reduce functions corresponding to the subset \mathcal{P} .

Given this partition, Node k , $k \in \{1, \dots, K\}$ computes the Reduce functions whose indices are in $\mathcal{D}_{\mathcal{P}}$ if $k \in \mathcal{P}$. Or equivalently, $\mathcal{D}_{\mathcal{P}} \subseteq \mathcal{W}_k$ if $k \in \mathcal{P}$. As a result, each node computes $\binom{K-1}{s-1} \eta_2 = \frac{sQ}{K}$ Reduce functions, i.e., $|\mathcal{W}_k| = \frac{sQ}{K}$ for all $k \in \{1, \dots, K\}$.

Example (Coded Distributed Computing: Reduce-Function Assignment). For our running example, the Reduce functions are assigned such that every $s = 2$ nodes compute a common Reduce function. More specifically as shown in Fig. 4, the sets of indices of the Reduce functions computed by the $K = 4$ nodes are $\mathcal{W}_1 = \{1, 2, 3\}$, $\mathcal{W}_2 = \{1, 4, 5\}$, $\mathcal{W}_3 = \{2, 4, 6\}$, and $\mathcal{W}_4 = \{3, 5, 6\}$. Therefore, for example, Node 1 still needs the intermediate

values $\{v_{q,n} : q \in \{1, 2, 3\}, n \in \{4, 5, 6\}\}$ through data shuffling to compute its assigned Reduce functions h_1, h_2, h_3 . \square

For a subset \mathcal{S} of $\{1, \dots, K\}$ and $\mathcal{S}_1 \subset \mathcal{S}$ with $|\mathcal{S}_1| = r$, we denote the set of intermediate values needed by *all* nodes in $\mathcal{S} \setminus \mathcal{S}_1$, *no* node outside \mathcal{S} , and known *exclusively* by nodes in \mathcal{S}_1 as $\mathcal{V}_{\mathcal{S}_1}^{\mathcal{S} \setminus \mathcal{S}_1}$. More formally:

$$\mathcal{V}_{\mathcal{S}_1}^{\mathcal{S} \setminus \mathcal{S}_1} \triangleq \{v_{q,n} : q \in \bigcap_{k \in \mathcal{S} \setminus \mathcal{S}_1} \mathcal{W}_k, q \notin \bigcup_{k \notin \mathcal{S}} \mathcal{W}_k, w_n \in \bigcap_{k \in \mathcal{S}_1} \mathcal{M}_k, w_n \notin \bigcup_{k \notin \mathcal{S}_1} \mathcal{M}_k\}. \quad (9)$$

The shuffling scheme of CDC consists of multiple rounds, each corresponding to all the subsets of the K nodes with a particular size. Within each subset, the needed intermediate values are split into *segments*, and the segments are uniformly associated to a subset of nodes that have computed the corresponding intermediate values locally. Then each node multicasts *random linear combinations* of the segments that are associated with it. More specifically, for all the subsets $\mathcal{S} \subseteq \{1, \dots, K\}$ of size $\max\{r+1, s\} \leq |\mathcal{S}| \leq \min\{r+s, K\}$ and all $\mathcal{S}_1 \subset \mathcal{S}$ of size $|\mathcal{S}_1| = r$:

- 1) For each intermediate value $v_{q,n}$ in $\mathcal{V}_{\mathcal{S}_1}^{\mathcal{S} \setminus \mathcal{S}_1}$, we evenly split it into r segments of $\frac{T}{r}$ bits: $v_{q,n}^{(1)}, \dots, v_{q,n}^{(r)}$, and associate each segment with a distinct node in \mathcal{S}_1 . By doing so we evenly split $\mathcal{V}_{\mathcal{S}_1}^{\mathcal{S} \setminus \mathcal{S}_1}$ into r disjoint partitions $\{\mathcal{V}_{\mathcal{S}_1,j}^{\mathcal{S} \setminus \mathcal{S}_1} : j \in \mathcal{S}_1\}$.
- 2) For each Node j in \mathcal{S} , it sends $\binom{|\mathcal{S}|-2}{r-1} \binom{r}{|\mathcal{S}|-s} \eta_1 \eta_2$ random linear combinations of all segments associated with it, which can be formally expressed as

$$\mathcal{U}_j^{\mathcal{S}} \triangleq \bigcup_{\mathcal{S}_1 \subset \mathcal{S}, |\mathcal{S}_1|=r, j \in \mathcal{S}_1} \mathcal{V}_{\mathcal{S}_1,j}^{\mathcal{S} \setminus \mathcal{S}_1}. \quad (10)$$

Remark 8. When $s = 1$, i.e., every output function is computed by one node, the above coded shuffling scheme only takes one round for all subsets \mathcal{S} of size $|\mathcal{S}| = r + 1$. Instead of sending random linear combinations as specified in Step 2, every node in \mathcal{S} can simply send the bit-wise XOR of its associated segments. \square

Example (Coded Distributed Computing: Coded Data Shuffling). Applying the above described shuffling scheme to the running example results in two rounds of coded data exchange, which are illustrated in Fig. 4. In the first round, intermediate values are communicated within each subset of 3 nodes.

In the subset $\mathcal{S} = \{1, 2, 3\}$, we have as defined in (9), $\mathcal{V}_{\{2,3\}}^{\{1\}} = \{v_{1,4}, v_{2,4}\}$, $\mathcal{V}_{\{1,3\}}^{\{2\}} = \{v_{1,2}, v_{4,2}\}$, and $\mathcal{V}_{\{1,2\}}^{\{3\}} = \{v_{2,1}, v_{4,1}\}$. We then associate the intermediate values to the nodes such that Node 1 is responsible for sending $\{v_{1,2}, v_{2,1}\}$, Node 2 is responsible for sending $\{v_{4,1}, v_{1,4}\}$, and Node 3 is responsible for sending $\{v_{4,2}, v_{2,4}\}$. During the data shuffling, each node in the set $\{1, 2, 3\}$ multicasts the bit-wise XOR, denoted by \oplus , of its associated intermediate values.

Since Node 2 knows $v_{2,1}$ and Node 3 knows $v_{1,2}$ from their locally computed Map functions, they can respectively decode $v_{1,2}$ and $v_{2,1}$ from the coded message $v_{1,2} \oplus v_{2,1}$.

We employ the similar shuffling scheme on the other 3 subsets of 3 nodes. After the first round of shuffling:

- Node 1 decodes $(v_{1,4}, v_{1,5})$, $(v_{2,4}, v_{2,6})$ and $(v_{3,5}, v_{3,6})$,
- Node 2 decodes $(v_{1,2}, v_{1,3})$, $(v_{4,2}, v_{4,6})$ and $(v_{5,3}, v_{5,6})$,
- Node 3 decodes $(v_{2,1}, v_{2,3})$, $(v_{4,1}, v_{4,5})$ and $(v_{6,3}, v_{6,5})$,
- Node 4 decodes $(v_{3,1}, v_{3,2})$, $(v_{5,1}, v_{5,4})$ and $(v_{6,2}, v_{6,4})$.

In the second round, we consider the set of all 4 nodes. We split each intermediate value $v_{q,n}$ in $\{\mathcal{V}_{S_1}^{\{1,2,3,4\}} \setminus S_1 : |S_1|=2\}$ into $r = 2$ segments $v_{q,n}^{(1)}$ and $v_{q,n}^{(2)}$, and associate them with the nodes such that Node 1 is responsible for sending $\{v_{4,3}^{(1)}, v_{5,2}^{(1)}, v_{6,1}^{(1)}\}$, Node 2 is responsible for sending $\{v_{2,5}^{(1)}, v_{3,4}^{(1)}, v_{6,1}^{(2)}\}$, Node 3 is responsible for sending $\{v_{1,6}^{(1)}, v_{3,4}^{(2)}, v_{5,2}^{(2)}\}$, and Node 4 is responsible for sending $\{v_{1,6}^{(2)}, v_{2,5}^{(2)}, v_{4,3}^{(2)}\}$. That is as defined in (10):

- $\mathcal{U}_1^{\{1,2,3,4\}} = \{v_{4,3}^{(1)}, v_{5,2}^{(1)}, v_{6,1}^{(1)}\}$,
- $\mathcal{U}_2^{\{1,2,3,4\}} = \{v_{2,5}^{(1)}, v_{3,4}^{(1)}, v_{6,1}^{(2)}\}$,
- $\mathcal{U}_3^{\{1,2,3,4\}} = \{v_{1,6}^{(1)}, v_{3,4}^{(2)}, v_{5,2}^{(2)}\}$,
- $\mathcal{U}_4^{\{1,2,3,4\}} = \{v_{1,6}^{(2)}, v_{2,5}^{(2)}, v_{4,3}^{(2)}\}$.

Then for each $k \in \{1, 2, 3, 4\}$, Node k multicasts 2 random linear combinations $C_k^1(\cdot, \cdot, \cdot), C_k^2(\cdot, \cdot, \cdot)$ of the 3 data segments in \mathcal{U}_k , which are $C_1^i(v_{4,3}^{(1)}, v_{5,2}^{(1)}, v_{6,1}^{(1)})$, $C_2^i(v_{2,5}^{(1)}, v_{3,4}^{(1)}, v_{6,1}^{(2)})$, $C_3^i(v_{1,6}^{(1)}, v_{3,4}^{(2)}, v_{5,2}^{(2)})$, $C_4^i(v_{1,6}^{(2)}, v_{2,5}^{(2)}, v_{4,3}^{(2)})$, for $i = 1, 2$.

Having received the 2 coded data segments $C_1^i(v_{4,3}^{(1)}, v_{5,2}^{(1)}, v_{6,1}^{(1)})$, $i = 1, 2$, Node 2 decodes $\{v_{4,3}^{(1)}, v_{5,2}^{(1)}\}$ by canceling $v_{6,1}^{(1)}$ that is known locally. Similarly, Node 3 decodes $\{v_{4,3}^{(1)}, v_{6,1}^{(1)}\}$, and Node 4 decodes $\{v_{5,2}^{(1)}, v_{6,1}^{(1)}\}$. After the second round of data shuffling:

- Node 1 decodes $v_{1,6}$, $v_{2,5}$ and $v_{3,4}$,
- Node 2 decodes $v_{1,6}$, $v_{4,3}$ and $v_{5,2}$,
- Node 3 decodes $v_{2,5}$, $v_{4,3}$ and $v_{6,1}$,
- Node 4 decodes $v_{3,4}$, $v_{5,2}$ and $v_{6,1}$.

After two rounds of coded data shuffling, it is not difficult to verify that every node decodes all the required input values for its assigned Reduce functions from the received coded messages, with the help of its locally computed Map functions. \square

C. Communication Load

In Step 2 of the above shuffling scheme, since each random linear combination of segments has $\frac{T}{r}$ bits, there are a total of $\frac{|\mathcal{S}|}{r} \binom{|\mathcal{S}|-2}{r-1} \binom{r}{|\mathcal{S}|-s} \eta_1 \eta_2 T$ bits communicated within \mathcal{S} . This is true for all subsets $\mathcal{S} \subseteq \{1, \dots, K\}$ of size $\max\{r+1, s\} \leq |\mathcal{S}| \leq \min\{r+s, K\}$, and thus communication load achieved by the CDC scheme is

$$L_{\text{coded}}(r, s) = \sum_{\ell=\max\{r+1, s\}}^{\min\{r+s, K\}} \frac{\binom{K}{\ell} \frac{\ell-2}{r-1} \binom{r}{\ell-s} \eta_1 \eta_2 T}{QNT} = \sum_{\ell=\max\{r+1, s\}}^{\min\{r+s, K\}} \frac{\ell \binom{K}{\ell} \binom{\ell-2}{r-1} \binom{r}{\ell-s}}{r \binom{K}{r} \binom{K}{s}}. \quad (11)$$

Example (Coded Distributed Computing Communication Load). The shuffling scheme of our running example has two rounds, first for the subsets of 3 nodes and second for the set of all 4 nodes. In the first round, each node sends 3 XORed intermediate values, each of T bits. In the second round, every node sends 2 random linear combinations of the segments, each of $\frac{T}{2}$ bits. Therefore, the communication load achieved by the CDC scheme is $\frac{4(3T+2 \cdot \frac{1}{2}T)}{QNT} = \frac{16}{6 \cdot 6} = \frac{4}{9}$, which matches the upper bound in Theorem 2 for $r = s = 2$. \square

D. Correctness of CDC

We demonstrate the correctness of the CDC scheme by showing 1) Each node can decode its needed intermediate values from the received random linear combinations of the segments. 2) After the Shuffle phase, every node has *all* the needed intermediate values for its assigned output functions.

We start by making the following observations:

- For each \mathcal{S} and $\mathcal{S}_1 \subset \mathcal{S}$ with $|\mathcal{S}_1| = r$, the set $\mathcal{V}_{\mathcal{S}_1}^{\mathcal{S} \setminus \mathcal{S}_1}$ defined in (9) contains intermediate values of $\binom{r}{|\mathcal{S}|-s} \eta_2$ output functions. This is because that the output functions whose intermediate values are included in $\mathcal{V}_{\mathcal{S}_1}^{\mathcal{S} \setminus \mathcal{S}_1}$ should be computed *exclusively* by the nodes in $\mathcal{S} \setminus \mathcal{S}_1$ and a subset of $s - (|\mathcal{S}| - r)$ nodes in \mathcal{S}_1 . Therefore, $\mathcal{V}_{\mathcal{S}_1}^{\mathcal{S} \setminus \mathcal{S}_1}$ contains the intermediate values of a total of $\binom{r}{s-(|\mathcal{S}|-r)} \eta_2 = \binom{r}{|\mathcal{S}|-s} \eta_2$ output functions. Since every subset of r nodes map a unique batch of η_1 files, the size of $\mathcal{V}_{\mathcal{S}_1}^{\mathcal{S} \setminus \mathcal{S}_1}$ is $|\mathcal{V}_{\mathcal{S}_1}^{\mathcal{S} \setminus \mathcal{S}_1}| = \binom{r}{|\mathcal{S}|-s} \eta_1 \eta_2 T$ bits.
- For every subset \mathcal{S} and Node $j \in \mathcal{S}$, since there are $\binom{|\mathcal{S}|-1}{r-1}$ subsets $\mathcal{S}_1 \subset \mathcal{S}$ of size r that contain j , and every intermediate value in $\{\mathcal{V}_{\mathcal{S}_1}^{\mathcal{S} \setminus \mathcal{S}_1} : \mathcal{S}_1 \subset \mathcal{S}, |\mathcal{S}_1| = r, j \in \mathcal{S}_1\}$ has a segment associated with Node j , $\mathcal{U}_j^{\mathcal{S}}$ defined in (10) contains $\binom{|\mathcal{S}|-1}{r-1} \binom{r}{|\mathcal{S}|-s} \eta_1 \eta_2$ segments of the intermediate values.

Now consider a pair of nodes in \mathcal{S} , say Node j and k . Out of all the data segments in $\mathcal{U}_j^{\mathcal{S}}$, $\binom{|\mathcal{S}|-2}{r-2} \binom{r}{|\mathcal{S}|-s} \eta_1 \eta_2$ of them are also known at Node k (since there are $\binom{|\mathcal{S}|-2}{r-2}$ size- r subsets of \mathcal{S} that contain both Node j and Node k), and the rest are needed by Node k . Given the above observation that $\mathcal{U}_j^{\mathcal{S}}$ contains $\binom{|\mathcal{S}|-1}{r-1} \binom{r}{|\mathcal{S}|-s} \eta_1 \eta_2$ segments, Node j sending $\left[\binom{|\mathcal{S}|-1}{r-1} - \binom{|\mathcal{S}|-2}{r-2} \right] \binom{r}{|\mathcal{S}|-s} \eta_1 \eta_2 = \binom{|\mathcal{S}|-2}{r-1} \binom{r}{|\mathcal{S}|-s} \eta_1 \eta_2$ random linear combinations of all the segments in $\mathcal{U}_j^{\mathcal{S}}$ (Step 2 of the above shuffling scheme) suffice to deliver the $\binom{|\mathcal{S}|-2}{r-1} \binom{r}{|\mathcal{S}|-s} \eta_1 \eta_2$ segments needed by Node k . Moreover, these linear combinations sent by Node j *simultaneously* deliver the segments needed by Node k' for all $k' \in \mathcal{S} \setminus \{j\}$, and this is true for all $j \in \mathcal{S}$.

To see how the CDC shuffling scheme delivers all required inputs of the Reduce functions, WLOG we assume that the Reduce function h_1 is to be computed by Node 1. Then Node 1 will need a total of $\binom{K-1}{r} \eta_1$ distinct intermediate values of the output function ϕ_1 from other nodes (it already knows $\frac{rN}{K} = N - \binom{K-1}{r} \eta_1$ intermediate values of ϕ_1 by mapping the files in \mathcal{M}_1). By the assignment of the Reduce functions, there exists a subset \mathcal{S}_2 of size s containing Node 1 such that all nodes in \mathcal{S}_2 need to compute h_1 . Then during the data shuffling for each subset \mathcal{S} containing \mathcal{S}_2 (note that by the definition of $\mathcal{V}_{\mathcal{S}_1}^{\mathcal{S} \setminus \mathcal{S}_1}$ in (9), the intermediate values of ϕ_1 will *not* be communicated to Node 1 if $\mathcal{S}_2 \not\subseteq \mathcal{S}$, and this is because that some node outside \mathcal{S} also wants to compute h_1), there are $\binom{s-1}{|\mathcal{S}|-r-1}$ subsets \mathcal{S}_1 of \mathcal{S} with size $|\mathcal{S}_1| = r$ such that $1 \notin \mathcal{S}_1$ and $\mathcal{S} \setminus \mathcal{S}_1 \subseteq \mathcal{S}_2$, and thus Node 1 decodes $\binom{s-1}{|\mathcal{S}|-r-1} \eta_1$ distinct intermediate values of ϕ_1 . Therefore, the total number of distinct intermediate values of ϕ_1 Node 1 decodes over the entire Shuffle phase is

$$\sum_{\ell=\max\{r+1,s\}}^{\min\{r+s,K\}} \binom{s-1}{\ell-r-1} \binom{K-s}{\ell-s} \eta_1 = \binom{K-1}{r} \eta_1, \quad (12)$$

which matches the required number of intermediate values.

E. Non-Integer Valued Computation Load

For non-integer valued computation load $r \geq 1$, we generalize the CDC scheme as follows. We first expand the computation load $r = \alpha r_1 + (1 - \alpha)r_2$ as a convex combination of $r_1 \triangleq \lfloor r \rfloor$ and $r_2 \triangleq \lceil r \rceil$, for some $0 \leq \alpha \leq 1$. Then we partition the set of N input files $\{w_1, \dots, w_N\}$ into two disjoint subsets \mathcal{I}_1 and \mathcal{I}_2 of sizes $|\mathcal{I}_1| = \alpha N$ and $|\mathcal{I}_2| = (1 - \alpha)N$. We next apply the CDC scheme described above respectively to the files in \mathcal{I}_1 with a computation load r_1 and the files in \mathcal{I}_2 with a computation load r_2 , to compute each of the Q output functions at the same set of s nodes. This results in a communication load of

$$\frac{Q\alpha N L_{\text{coded}}(r_1, s)T + Q(1 - \alpha)N L_{\text{coded}}(r_2, s)T}{QNT} = \alpha L_{\text{coded}}(r_1, s) + (1 - \alpha)L_{\text{coded}}(r_2, s), \quad (13)$$

where $L_{\text{coded}}(r, s)$ is the communication load achieved by CDC in (11) for integer-valued $r, s \in \{1, \dots, K\}$.

Using this generalized CDC scheme, for any two integer-valued computation loads r_1 and r_2 , the points on the line segment connecting $(r_1, L_{\text{coded}}(r_1, s))$ and $(r_2, L_{\text{coded}}(r_2, s))$ are achievable. Therefore, for general $1 \leq r \leq K$, the *lower convex envelop* of the achievable points $\{(r, L_{\text{coded}}(r, s)) : r \in \{1, \dots, K\}\}$ is achievable. This proves the upper bound on the computation-communication function in Theorem 2 (also the achievability part of Theorem 1 by setting $s = 1$).

Remark 9. The ideas of efficiently creating and exploiting coded multicasting opportunities have been introduced in caching problems [16]–[18]. This type of coding was also utilized to solve the index coding problem [22], [23] that arises from the network coding problem [24].

In this paper, we propose a distributed computing framework, and characterize the optimal tradeoff between computation and communication in the proposed framework, which is achieved by the proposed Coded Distributed Computing (CDC) scheme. For the case of $s = 1$ where no two nodes are interested in computing a common Reduce function, the coded data shuffling of CDC is similar to a coded transmission strategy in wireless D2D networks proposed in [18], where the “side information” enabling coded multicasting are pre-fetched in a specific repetitive manner in the caches of wireless nodes (in CDC such information is obtained by computing the Map functions locally). When s is larger than 1, i.e., every Reduce function needs to be computed at multiple nodes, CDC creates novel coding opportunities that exploit both the redundancy of the Map computations and the commonality of the data requests for Reduce functions across nodes, further reducing the communication load. \square

V. CONVERSE OF THEOREM 1

In this section, we prove the lower bound on $L^*(r)$ in Theorem 1.

For $k \in \{1, \dots, K\}$, we denote the set of indices of the files mapped by Node k as \mathcal{M}_k , and the set of indices of the Reduce functions computed by Node k as \mathcal{W}_k . As the first step, we consider the communication load for a given file assignment $\mathcal{M} \triangleq \{\mathcal{M}_k\}_{k=1}^K$ in the Map phase. We denote the minimum communication load under the file assignment \mathcal{M} by $L_{\mathcal{M}}^*$.

We denote the number of files that are mapped at j nodes under a file assignment \mathcal{M} , as $a_{\mathcal{M}}^j$, for all $j \in \{1, \dots, K\}$:

$$a_{\mathcal{M}}^j = \sum_{\mathcal{J} \subseteq \{1, \dots, K\}: |\mathcal{J}|=j} |(\cap_{k \in \mathcal{J}} \mathcal{M}_k) \setminus (\cup_{i \notin \mathcal{J}} \mathcal{M}_i)|. \quad (14)$$

For a particular file assignment \mathcal{M} , we present a lower bound on $L_{\mathcal{M}}^*$ in the following lemma.

Lemma 1. $L_{\mathcal{M}}^* \geq \sum_{j=1}^K \frac{a_{\mathcal{M}}^j}{N} \cdot \frac{K-j}{Kj}$.

Next, we first demonstrate the converse of Theorem 1 using Lemma 1, and then give the proof of Lemma 1.

Converse Proof of Theorem 1. It is clear that the minimum communication load $L^*(r)$ is lower bounded by the minimum value of $L_{\mathcal{M}}^*$ over all possible file assignments which admit a computation load of r :

$$L^*(r) \geq \inf_{\mathcal{M}: |\mathcal{M}_1| + \dots + |\mathcal{M}_K| = rN} L_{\mathcal{M}}^*. \quad (15)$$

Then by Lemma 1, we have

$$L^*(r) \geq \inf_{\mathcal{M}: |\mathcal{M}_1| + \dots + |\mathcal{M}_K| = rN} \sum_{j=1}^K \frac{a_{\mathcal{M}}^j}{N} \cdot \frac{K-j}{Kj}. \quad (16)$$

For every file assignment \mathcal{M} such that $|\mathcal{M}_1| + \dots + |\mathcal{M}_K| = rN$, $\{a_{\mathcal{M}}^j\}_{j=1}^K$ satisfy

$$a_{\mathcal{M}}^j \geq 0, \quad j \in \{1, \dots, K\}, \quad (17)$$

$$\sum_{j=1}^K a_{\mathcal{M}}^j = N, \quad (18)$$

$$\sum_{j=1}^K j a_{\mathcal{M}}^j = rN. \quad (19)$$

Then since the function $\frac{K-j}{Kj}$ in (16) is convex in j , and by (18) $\sum_{j=1}^K \frac{a_{\mathcal{M}}^j}{N} = 1$, (16) becomes

$$L^*(r) \geq \inf_{\mathcal{M}: |\mathcal{M}_1| + \dots + |\mathcal{M}_K| = rN} \frac{K - \sum_{j=1}^K j \frac{a_{\mathcal{M}}^j}{N}}{K \sum_{j=1}^K j \frac{a_{\mathcal{M}}^j}{N}} \stackrel{(a)}{=} \frac{K-r}{Kr}, \quad (20)$$

where (a) is due to the requirement imposed by the computation load in (19).

The lower bound on $L^*(r)$ in (20) holds for general $1 \leq r \leq K$. We can further improve the lower bound for non-integer valued r as follows. For $r \notin \mathbb{N}$, we first find the line $p + qj$ connecting the two points $(\lfloor r \rfloor, \frac{K-\lfloor r \rfloor}{K\lfloor r \rfloor})$ and $(\lceil r \rceil, \frac{K-\lceil r \rceil}{K\lceil r \rceil})$, for some $p, q \in \mathbb{R}$. Then by the convexity of the function $\frac{K-j}{Kj}$, for all $j \in \{1, \dots, K\}$,

$$\frac{K-j}{Kj} \geq p + qj. \quad (21)$$

Then (16) reduces to

$$L^*(r) \geq \inf_{\mathcal{M}: |\mathcal{M}_1| + \dots + |\mathcal{M}_K| = rN} \sum_{j=1}^K \frac{a_{\mathcal{M}}^j}{N} \cdot (p + qj) \quad (22)$$

$$= \inf_{\mathcal{M}: |\mathcal{M}_1| + \dots + |\mathcal{M}_K| = rN} \sum_{j=1}^K \frac{a_{\mathcal{M}}^j}{N} \cdot p + \sum_{j=1}^K \frac{ja_{\mathcal{M}}^j}{N} \cdot q \quad (23)$$

$$\stackrel{(b)}{=} p + qr, \quad (24)$$

where (b) is due to the constraints on $\{a_{\mathcal{M}}^j\}_{j=1}^K$ in (18) and (19).

Therefore, $L^*(r)$ is lower bounded by the lower convex envelop of the points $\{(r, \frac{K-r}{Kr}) : r \in \{1, \dots, K\}\}$. This completes the proof of the converse part of Theorem 1. \blacksquare

Remark 10. Although the model proposed in this paper only allows each node sending messages independently, we can show that even if the data shuffling process can be carried out in multiple rounds and dependency between messages are allowed, the lower bound on $L^*(r)$ remains the same. \square

We devote the rest of this section to the proof of Lemma 1. To prove Lemma 1, we develop a lower bound on the number of bits communicated by any subset of nodes, by induction on the size of the subset. In particular, for a subset of computing nodes, we first characterize the minimum number of bits required by one of the nodes in the subset, in order to recover the intermediate values it needs for its assigned Reduce functions, and then combine this result with the lower bound on the number of bits communicated by the rest of the nodes in that subset, which is given by the inductive argument.

Proof of Lemma 1. For $q \in \{1, \dots, Q\}$, $n \in \{1, \dots, N\}$, we let $V_{q,n}$ be i.i.d. random variables uniformly distributed on \mathbb{F}_{2^T} . We let the intermediate values $v_{q,n}$ be the realizations of $V_{q,n}$. For some $\mathcal{Q} \subseteq \{1, \dots, Q\}$ and $\mathcal{N} \subseteq \{1, \dots, N\}$, we define

$$V_{\mathcal{Q}, \mathcal{N}} \triangleq \{V_{q,n} : q \in \mathcal{Q}, n \in \mathcal{N}\}. \quad (25)$$

Since each message X_k is generated as a function of the intermediate values that are computed at Node k , the following equation holds for all $k \in \{1, \dots, K\}$:²

$$H(X_k | V_{:, \mathcal{M}_k}) = 0. \quad (26)$$

The validity of the shuffling scheme requires that for all $k \in \{1, \dots, K\}$, the following equation holds :

$$H(V_{\mathcal{W}_k, :} | X_k, V_{:, \mathcal{M}_k}) = 0. \quad (27)$$

For a subset $\mathcal{S} \subseteq \{1, \dots, K\}$, let $\mathcal{S}^c = \{1, \dots, K\} \setminus \mathcal{S}$, and we define

$$Y_{\mathcal{S}^c} \triangleq (V_{\mathcal{W}_{\mathcal{S}^c}, :}, V_{:, \mathcal{M}_{\mathcal{S}^c}}). \quad (28)$$

²By an abuse of notation, we use “:” to denote the set of all possible indices.

For a file assignment \mathcal{M} , we denote the number of files that are *exclusively* mapped by j nodes in \mathcal{S} as $a_{\mathcal{M}}^{j,\mathcal{S}}$:

$$a_{\mathcal{M}}^{j,\mathcal{S}} = \sum_{\mathcal{J} \subseteq \mathcal{S}: |\mathcal{J}|=j} |(\cap_{k \in \mathcal{J}} \mathcal{M}_k) \setminus (\cup_{i \notin \mathcal{J}} \mathcal{M}_i)|, \quad (29)$$

Then we prove the following statement by induction:

Claim 1. For any subset $\mathcal{S} \subseteq \{1, \dots, K\}$, we have

$$H(X_{\mathcal{S}}|Y_{\mathcal{S}^c}) \geq T \sum_{j=1}^{|\mathcal{S}|} a_{\mathcal{M}}^{j,\mathcal{S}} \frac{Q}{K} \cdot \frac{|\mathcal{S}| - j}{j}. \quad (30)$$

□

a. If $\mathcal{S} = \emptyset$, obviously

$$H(X_{\emptyset}|Y_{\emptyset^c}) \geq 0 = T \sum_{j=1}^0 a_{\mathcal{M}}^{j,\emptyset} \frac{Q}{K} \cdot \frac{0 - j}{j}. \quad (31)$$

b. Suppose the statement is true for all subsets of size S_0 .

For any $\mathcal{S} \subseteq \{1, \dots, K\}$ of size $|\mathcal{S}| = S_0 + 1$, and all $k \in \mathcal{S}$, the subset version of (26) and (27) can be derived:

$$H(X_k|V_{:, \mathcal{M}_k}, Y_{\mathcal{S}^c}) = 0, \quad (32)$$

$$H(V_{\mathcal{W}_k, :}|X_{\mathcal{S}}, V_{:, \mathcal{M}_k}, Y_{\mathcal{S}^c}) = 0. \quad (33)$$

Consequently, the following equation holds:

$$H(X_{\mathcal{S}}|V_{:, \mathcal{M}_k}, Y_{\mathcal{S}^c}) = H(X_{\mathcal{S}}|V_{\mathcal{W}_k, :}, V_{:, \mathcal{M}_k}, Y_{\mathcal{S}^c}) + H(V_{\mathcal{W}_k, :}|V_{:, \mathcal{M}_k}, Y_{\mathcal{S}^c}). \quad (34)$$

Next we lower bound $H(X_{\mathcal{S}}|Y_{\mathcal{S}^c})$ as follows:

$$H(X_{\mathcal{S}}|Y_{\mathcal{S}^c}) = \frac{1}{|\mathcal{S}|} \sum_{k \in \mathcal{S}} H(X_{\mathcal{S}}, X_k|Y_{\mathcal{S}^c}) \quad (35)$$

$$= \frac{1}{|\mathcal{S}|} \sum_{k \in \mathcal{S}} (H(X_{\mathcal{S}}|X_k, Y_{\mathcal{S}^c}) + H(X_k|Y_{\mathcal{S}^c})) \quad (36)$$

$$\geq \frac{1}{|\mathcal{S}|} \sum_{k \in \mathcal{S}} H(X_{\mathcal{S}}|X_k, Y_{\mathcal{S}^c}) + \frac{1}{|\mathcal{S}|} H(X_{\mathcal{S}}|Y_{\mathcal{S}^c}). \quad (37)$$

From (37), we can derive a lower bound on $H(X_{\mathcal{S}}|Y_{\mathcal{S}^c})$ that equals the LHS of (34) scaled by $\frac{1}{S_0}$:

$$H(X_{\mathcal{S}}|Y_{\mathcal{S}^c}) \geq \frac{1}{|\mathcal{S}| - 1} \sum_{k \in \mathcal{S}} H(X_{\mathcal{S}}|X_k, Y_{\mathcal{S}^c}) \quad (38)$$

$$\geq \frac{1}{S_0} \sum_{k \in \mathcal{S}} H(X_{\mathcal{S}}|X_k, V_{:, \mathcal{M}_k}, Y_{\mathcal{S}^c}) \quad (39)$$

$$= \frac{1}{S_0} \sum_{k \in \mathcal{S}} H(X_{\mathcal{S}}|V_{:, \mathcal{M}_k}, Y_{\mathcal{S}^c}). \quad (40)$$

The first term on the RHS of (34) is lower bounded by the induction assumption:

$$H(X_{\mathcal{S}}|V_{\mathcal{W}_k, :}, V_{:, \mathcal{M}_k}, Y_{\mathcal{S}^c}) = H(X_{\mathcal{S} \setminus \{k\}}|Y_{(\mathcal{S} \setminus \{k\})^c}) \quad (41)$$

$$\geq T \sum_{j=1}^{S_0} a_{\mathcal{M}}^{j, \mathcal{S} \setminus \{k\}} \frac{Q}{K} \cdot \frac{S_0 - j}{j}. \quad (42)$$

The second term on the RHS of (34) can be calculated based on the independence of intermediate values:

$$H(V_{\mathcal{W}_k,:} | V_{:, \mathcal{M}_k}, Y_{\mathcal{S}^c}) = H(V_{\mathcal{W}_k,:} | V_{:, \mathcal{M}_k}, V_{\mathcal{W}_{\mathcal{S}^c},:}, V_{:, \mathcal{M}_{\mathcal{S}^c}}) \quad (43)$$

$$= T \frac{Q}{K} \sum_{j=0}^{S_0} a_{\mathcal{M}}^{j, \mathcal{S} \setminus \{k\}} \quad (44)$$

$$\geq T \frac{Q}{K} \sum_{j=1}^{S_0} a_{\mathcal{M}}^{j, \mathcal{S} \setminus \{k\}}. \quad (45)$$

Thus by (34), (40), (42) and (45), we have

$$H(X_{\mathcal{S}} | Y_{\mathcal{S}^c}) \geq \frac{1}{S_0} \sum_{k \in \mathcal{S}} H(X_{\mathcal{S}} | V_{:, \mathcal{M}_k}, Y_{\mathcal{S}^c}) \quad (46)$$

$$= \frac{1}{S_0} \sum_{k \in \mathcal{S}} (H(X_{\mathcal{S}} | V_{\mathcal{W}_k,:}, V_{:, \mathcal{M}_k}, Y_{\mathcal{S}^c}) + H(V_{\mathcal{W}_k,:} | V_{:, \mathcal{M}_k}, Y_{\mathcal{S}^c})) \quad (47)$$

$$\geq \frac{T}{S_0} \sum_{k \in \mathcal{S}} \sum_{j=1}^{S_0} a_{\mathcal{M}}^{j, \mathcal{S} \setminus \{k\}} \frac{Q}{K} \cdot \frac{S_0}{j} \quad (48)$$

$$= T \sum_{j=1}^{S_0} \frac{Q}{K} \cdot \frac{1}{j} \sum_{k \in \mathcal{S}} a_{\mathcal{M}}^{j, \mathcal{S} \setminus \{k\}}. \quad (49)$$

By the definition of $a_{\mathcal{M}}^{j, \mathcal{S}}$, we have the following equations:

$$\sum_{k \in \mathcal{S}} a_{\mathcal{M}}^{j, \mathcal{S} \setminus \{k\}} = \sum_{k \in \mathcal{S}} \sum_{f=1}^N \mathbb{1}(\text{file } f \text{ is only mapped by nodes in } \mathcal{S} \setminus \{k\}) \cdot \mathbb{1}(f \text{ is mapped by } j \text{ nodes}) \quad (50)$$

$$= \sum_{f=1}^N \mathbb{1}(\text{file } f \text{ is only mapped by } j \text{ nodes in } \mathcal{S}) \sum_{k \in \mathcal{S}} \mathbb{1}(f \text{ is not mapped by Node } k) \quad (51)$$

$$= \sum_{f=1}^N \mathbb{1}(\text{file } f \text{ is only mapped by } j \text{ nodes in } \mathcal{S}) (|\mathcal{S}| - j) \quad (52)$$

$$= a_{\mathcal{M}}^{j, \mathcal{S}} (S_0 + 1 - j). \quad (53)$$

Applying (53) to (49) yields

$$H(X_{\mathcal{S}} | Y_{\mathcal{S}^c}) \geq T \sum_{j=1}^{S_0} a_{\mathcal{M}}^{j, \mathcal{S}} \frac{Q}{K} \cdot \frac{S_0 + 1 - j}{j} \quad (54)$$

$$= T \sum_{j=1}^{S_0+1} a_{\mathcal{M}}^{j, \mathcal{S}} \frac{Q}{K} \cdot \frac{S_0 + 1 - j}{j}. \quad (55)$$

c. Thus for all subsets $\mathcal{S} \subseteq \{1, \dots, K\}$, the following equation holds:

$$H(X_{\mathcal{S}} | Y_{\mathcal{S}^c}) \geq T \sum_{j=1}^{|\mathcal{S}|} a_{\mathcal{M}}^{j, \mathcal{S}} \frac{Q}{K} \cdot \frac{|\mathcal{S}| - j}{j}, \quad (56)$$

which proves Claim 1.

Then by Claim 1, let $\mathcal{S} = \{1, \dots, K\}$ be the set of all K nodes,

$$L_{\mathcal{M}}^* \geq \frac{H(X_{\mathcal{S}}|Y_{\mathcal{S}^c})}{QNT} \geq \sum_{j=1}^K \frac{a_{\mathcal{M}}^j}{N} \cdot \frac{K-j}{Kj}. \quad (57)$$

This completes the proof of Lemma 1. ■

VI. CONVERSE OF THEOREM 2

In this section, we prove the lower bound on $L^*(r, s)$ in Theorem 2, which generalizes the converse result of Theorem 1 for the case $s > 1$.

We denote the minimum communication load under a particular file assignment \mathcal{M} as $L_{\mathcal{M}}^*(s)$, and we present a lower bound on $L_{\mathcal{M}}^*(s)$ in the following lemma.

Lemma 2. $L_{\mathcal{M}}^*(s) \geq \sum_{j=1}^K \frac{a_{\mathcal{M}}^j}{N} \sum_{\ell=\max\{j,s\}}^{\min\{j+s,K\}} \frac{\binom{K-j}{\ell-j} \binom{j}{\ell-s}}{\binom{K}{s}} \cdot \frac{\ell-j}{\ell-1}$, where $a_{\mathcal{M}}^j$ is defined in (14).

In the rest of this section, we first prove the converse part of Theorem 2, and then give the proof of Lemma 2.

We have shown in Section IV that for a computation load $r \in \{1, \dots, K\}$, the proposed CDC scheme achieves the communication load

$$L_{\text{coded}}(r, s) = \sum_{\ell=\max\{r+1,s\}}^{\min\{r+s,K\}} \frac{\ell \binom{K}{\ell} \binom{\ell-2}{r-1} \binom{r}{\ell-s}}{r \binom{K}{r} \binom{K}{s}} = \sum_{\ell=\max\{r,s\}}^{\min\{r+s,K\}} \frac{\binom{K-r}{\ell-r} \binom{r}{\ell-s}}{\binom{K}{s}} \cdot \frac{\ell-r}{\ell-1}. \quad (58)$$

Next, we utilize Lemma 2 to prove the converse of Theorem 2 by showing $L^*(r, s) \geq \sum_{\ell=\max\{r,s\}}^{\min\{r+s,K\}} \frac{\binom{K-r}{\ell-r} \binom{r}{\ell-s}}{\binom{K}{s}} \cdot \frac{\ell-r}{\ell-1}$.

Converse Proof of Theorem 2. The minimum communication load $L^*(r, s)$ is lower bounded by the minimum value of $L_{\mathcal{M}}^*(s)$ over all possible file assignments having a computation load of r :

$$L^*(r, s) \geq \inf_{\mathcal{M}: |\mathcal{M}_1| + \dots + |\mathcal{M}_K| = rN} L_{\mathcal{M}}^*(s). \quad (59)$$

Then by Lemma 2, we have

$$L^*(r, s) \geq \inf_{\mathcal{M}: |\mathcal{M}_1| + \dots + |\mathcal{M}_K| = rN} \sum_{j=1}^K \frac{a_{\mathcal{M}}^j}{N} \sum_{\ell=\max\{j,s\}}^{\min\{j+s,K\}} \frac{\binom{K-j}{\ell-j} \binom{j}{\ell-s}}{\binom{K}{s}} \cdot \frac{\ell-j}{\ell-1}. \quad (60)$$

For every file assignment \mathcal{M} such that $|\mathcal{M}_1| + \dots + |\mathcal{M}_K| = rN$, $\{a_{\mathcal{M}}^j\}_{j=1}^K$ satisfy the same conditions as the case of $s = 1$ in (17), (18) and (19).

For a general computation load $1 \leq r \leq K$, we first find the line $p + qj$ connecting the two points $(\lfloor r \rfloor, L_{\text{coded}}(\lfloor r \rfloor, s))$ and $(\lceil r \rceil, L_{\text{coded}}(\lceil r \rceil, s))$, for some $p, q \in \mathbb{R}$. Then by the convexity of the function $L_{\text{coded}}(j, s)$ in j , for all $j \in \{1, \dots, K\}$,

$$L_{\text{coded}}(j, s) = \sum_{\ell=\max\{j,s\}}^{\min\{j+s,K\}} \frac{\binom{K-j}{\ell-j} \binom{j}{\ell-s}}{\binom{K}{s}} \cdot \frac{\ell-j}{\ell-1} \geq p + qj. \quad (61)$$

Then (60) reduces to

$$L^*(r, s) \geq \inf_{\mathcal{M}: |\mathcal{M}_1| + \dots + |\mathcal{M}_K| = rN} \sum_{j=1}^K \frac{a_{\mathcal{M}}^j}{N} \cdot (p + qj) \quad (62)$$

$$= \inf_{\mathcal{M}: |\mathcal{M}_1| + \dots + |\mathcal{M}_K| = rN} \sum_{j=1}^K \frac{a_{\mathcal{M}}^j}{N} \cdot p + \sum_{j=1}^K \frac{j a_{\mathcal{M}}^j}{N} \cdot q \quad (63)$$

$$\stackrel{(a)}{=} p + qr, \quad (64)$$

where (a) is due to the constraints on $\{a_{\mathcal{M}}^j\}_{j=1}^K$ in (18) and (19).

Therefore, $L^*(r, s)$ is lower bounded by the lower convex envelop of the points $\{(r, L_{\text{coded}}(r, s)) : r \in \{1, \dots, K\}\}$.

This completes the proof of the converse part of Theorem 2. \blacksquare

The proof of lemma 2 follows the same steps of the proof of Lemma 1, where a lower bound on the number of bits communicated by any subset of nodes, for the general case of $s \geq 1$, is established by induction.

Proof of Lemma 2. We state and prove the following statement by induction:

Claim 2. For any subset $\mathcal{S} \subseteq \{1, \dots, K\}$, we have

$$H(X_{\mathcal{S}}|Y_{\mathcal{S}^c}) \geq QT \sum_{j=1}^{|\mathcal{S}|} a_{\mathcal{M}}^{j, \mathcal{S}} \sum_{\ell=\max\{j, s\}}^{\min\{j+s, |\mathcal{S}|\}} \frac{\binom{|\mathcal{S}|-j}{\ell-j} \binom{j}{\ell-s}}{\binom{K}{s}} \cdot \frac{\ell-j}{\ell-1}, \quad (65)$$

where $a_{\mathcal{M}}^{j, \mathcal{S}}$ is defined in (29). \square

a. If $\mathcal{S} = \emptyset$, obviously

$$H(X_{\emptyset}|Y_{\emptyset^c}) \geq 0 = QT \sum_{j=1}^0 a_{\mathcal{M}}^{j, \emptyset} \sum_{\ell=\max\{j, s\}}^{\min\{j+s, 0\}} \frac{\binom{0-j}{\ell-j} \binom{j}{\ell-s}}{\binom{K}{s}} \cdot \frac{\ell-j}{\ell-1}. \quad (66)$$

b. Suppose the statement is true for all subsets of size S_0 .

For any $\mathcal{S} \subseteq \{1, \dots, K\}$ of size $|\mathcal{S}| = S_0 + 1$, and all $k \in \mathcal{S}$, we have as derived in (47):

$$H(X_{\mathcal{S}}|Y_{\mathcal{S}^c}) \geq \frac{1}{S_0} \sum_{k \in \mathcal{S}} (H(X_{\mathcal{S}}|V_{\mathcal{W}_k, :}, V_{:, \mathcal{M}_k}, Y_{\mathcal{S}^c}) + H(V_{\mathcal{W}_k, :}|V_{:, \mathcal{M}_k}, Y_{\mathcal{S}^c})), \quad (67)$$

where $Y_{\mathcal{S}^c}$ is defined in (28).

The first term on the RHS of (67) is lower bounded by the induction assumption:

$$H(X_{\mathcal{S}}|V_{\mathcal{W}_k, :}, V_{:, \mathcal{M}_k}, Y_{\mathcal{S}^c}) = H(X_{\mathcal{S} \setminus \{k\}}|Y_{(\mathcal{S} \setminus \{k\})^c}) \quad (68)$$

$$\geq QT \sum_{j=1}^{S_0} a_{\mathcal{M}}^{j, \mathcal{S} \setminus \{k\}} \sum_{\ell=\max\{j, s\}}^{\min\{j+s, S_0\}} \frac{\binom{S_0-j}{\ell-j} \binom{j}{\ell-s}}{\binom{K}{s}} \cdot \frac{\ell-j}{\ell-1}. \quad (69)$$

The second term on the RHS of (67) can be calculated based on the independence of intermediate values:

$$H(V_{\mathcal{W}_k, :}|V_{:, \mathcal{M}_k}, Y_{\mathcal{S}^c}) \stackrel{(a)}{=} QT \frac{\binom{|\mathcal{S}|-1}{s-1}}{\binom{K}{s}} \sum_{j=0}^{S_0} a_{\mathcal{M}}^{j, \mathcal{S} \setminus \{k\}} \quad (70)$$

$$\geq QT \frac{\binom{S_0}{s-1}}{\binom{K}{s}} \sum_{j=1}^{S_0} a_{\mathcal{M}}^{j, \mathcal{S} \setminus \{k\}}, \quad (71)$$

where (a) is due to the uniform distribution of the output functions such that each node in \mathcal{S} calculates $\frac{Q}{\binom{K}{s}} \cdot \binom{|\mathcal{S}|-1}{s-1}$ output functions computed exclusively by s nodes in \mathcal{S} .

Thus by (67), (69) and (71), we have

$$H(X_S|Y_{S^c}) \geq \frac{QT}{S_0} \sum_{k \in \mathcal{S}} \sum_{j=1}^{S_0} a_{\mathcal{M}}^{j, \mathcal{S} \setminus \{k\}} \left(\sum_{\ell=\max\{j,s\}}^{\min\{j+s, S_0\}} \frac{\binom{S_0-j}{\ell-j} \binom{j}{\ell-s}}{\binom{K}{s}} \cdot \frac{\ell-j}{\ell-1} + \frac{\binom{S_0}{s-1}}{\binom{K}{s}} \right) \quad (72)$$

$$= \frac{QT}{S_0} \sum_{j=1}^{S_0} \left(\sum_{\ell=\max\{j,s\}}^{\min\{j+s, S_0\}} \frac{\binom{S_0-j}{\ell-j} \binom{j}{\ell-s}}{\binom{K}{s}} \cdot \frac{\ell-j}{\ell-1} + \frac{\binom{S_0}{s-1}}{\binom{K}{s}} \right) \sum_{k \in \mathcal{S}} a_{\mathcal{M}}^{j, \mathcal{S} \setminus \{k\}} \quad (73)$$

$$= QT \cdot \frac{S_0+1-j}{S_0} \sum_{j=1}^{S_0} \left(\sum_{\ell=\max\{j,s\}}^{\min\{j+s, S_0\}} \frac{\binom{S_0-j}{\ell-j} \binom{j}{\ell-s}}{\binom{K}{s}} \cdot \frac{\ell-j}{\ell-1} + \frac{\binom{S_0}{s-1}}{\binom{K}{s}} \right) a_{\mathcal{M}}^{j, \mathcal{S}} \quad (74)$$

$$= QT \sum_{j=1}^{S_0+1} \frac{S_0+1-j}{S_0} \left(\sum_{\ell=\max\{j,s\}}^{\min\{j+s, S_0\}} \frac{\binom{S_0-j}{\ell-j} \binom{j}{\ell-s}}{\binom{K}{s}} \cdot \frac{\ell-j}{\ell-1} + \frac{\binom{S_0}{s-1}}{\binom{K}{s}} \right) a_{\mathcal{M}}^{j, \mathcal{S}}. \quad (75)$$

For each $j \in \{1, \dots, S_0+1\}$ in (75), we have

$$\begin{aligned} & \frac{S_0+1-j}{S_0} \left(\sum_{\ell=\max\{j,s\}}^{\min\{j+s, S_0\}} \frac{\binom{S_0-j}{\ell-j} \binom{j}{\ell-s}}{\binom{K}{s}} \cdot \frac{\ell-j}{\ell-1} + \frac{\binom{S_0}{s-1}}{\binom{K}{s}} \right) \\ &= \frac{S_0+1-j}{S_0 \binom{K}{s}} \left(\sum_{\ell=\max\{j,s\}}^{\min\{j+s, S_0\}} \binom{S_0-j}{\ell-j} \binom{j}{\ell-s} \frac{\ell-j}{\ell-1} + \sum_{\ell=\max\{j+1, s\}}^{\min\{j+s, S_0+1\}} \binom{S_0-j}{\ell-j-1} \binom{j}{\ell-s} \right) \end{aligned} \quad (76)$$

$$= \frac{S_0+1-j}{S_0 \binom{K}{s}} \left(\sum_{\ell=\max\{j,s\}}^{\min\{j+s, S_0+1\}} \binom{S_0-j}{\ell-j} \binom{j}{\ell-s} \frac{\ell-j}{\ell-1} + \sum_{\ell=\max\{j, s\}}^{\min\{j+s, S_0+1\}} \binom{S_0-j}{\ell-j-1} \binom{j}{\ell-s} \right) \quad (77)$$

$$= \frac{1}{\binom{K}{s}} \sum_{\ell=\max\{j,s\}}^{\min\{j+s, S_0+1\}} \binom{S_0+1-j}{\ell-j} \binom{j}{\ell-s} \left(\frac{S_0-\ell+1}{S_0} \cdot \frac{\ell-j}{\ell-1} + \frac{\ell-j}{S_0} \right) \quad (78)$$

$$= \frac{1}{\binom{K}{s}} \sum_{\ell=\max\{j,s\}}^{\min\{j+s, S_0+1\}} \binom{S_0+1-j}{\ell-j} \binom{j}{\ell-s} \frac{\ell-j}{\ell-1}. \quad (79)$$

Applying (79) to (75) yields

$$H(X_S|Y_{S^c}) \geq QT \sum_{j=1}^{S_0+1} a_{\mathcal{M}}^{j, \mathcal{S}} \sum_{\ell=\max\{j,s\}}^{\min\{j+s, S_0+1\}} \frac{\binom{S_0+1-j}{\ell-j} \binom{j}{\ell-s}}{\binom{K}{s}} \cdot \frac{\ell-j}{\ell-1} \quad (80)$$

$$= QT \sum_{j=1}^{|\mathcal{S}|} a_{\mathcal{M}}^{j, \mathcal{S}} \sum_{\ell=\max\{j,s\}}^{\min\{j+s, |\mathcal{S}|\}} \frac{\binom{|\mathcal{S}|-j}{\ell-j} \binom{j}{\ell-s}}{\binom{K}{s}} \cdot \frac{\ell-j}{\ell-1}. \quad (81)$$

Since (81) holds for all subsets \mathcal{S} of size $|\mathcal{S}| = S_0+1$, we have proven Claim 2.

Then by Claim 2, let $\mathcal{S} = \{1, \dots, K\}$ be the set of all K nodes,

$$L_{\mathcal{M}}^*(s) \geq \frac{H(X_S|Y_{S^c})}{QNT} \geq \sum_{j=1}^K \frac{a_{\mathcal{M}}^j}{N} \sum_{\ell=\max\{j,s\}}^{\min\{j+s, K\}} \frac{\binom{K-j}{\ell-j} \binom{j}{\ell-s}}{\binom{K}{s}} \cdot \frac{\ell-j}{\ell-1}. \quad (82)$$

This completes the proof of Lemma 2. ■

VII. PRACTICAL CHALLENGES

In this section, we discuss and address some of the practical challenges in implementing the Coded Distributed Computing (CDC) scheme in current systems. In particular, we focus on the case where every output functions is computed by one node ($s = 1$).

A. Multicast Capability

In the development of CDC, we assumed the capability of network-layer multicast, which is however, often disabled in nowadays data center networks like Amazon EC2 clusters. One reason for not enabling multicasting is that there is often no opportunity for it, and therefore, the gain of multicasting would not be considerable in overall performance of the system. In contrary, in this paper, we show that by following a particular pattern in Map assignment, and using coding, one can *create* multicasting opportunities that scale with the size of the system, and significantly reduce the communication load of the network.

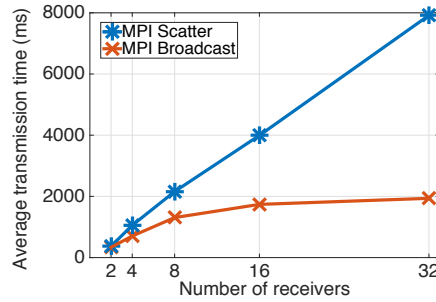


Fig. 5: Comparison of the average transmission time to communicate 4.15 MB, measured in [20], using Message Passing Interface (MPI) scatter (unicast) to individual receivers with that using MPI broadcast.

Although network-layer multicast is often not available in current systems, we can instead utilize existing application-layer multicast algorithms over underlying unicast networks. For example using the Message Passing Interface (MPI), as demonstrated in Fig. 5 (Fig. 14 in [20]), while the average transmission time of unicasting (scatter) to individual receivers increases linearly with the number of receivers, the broadcast API `MPI_Bcast` uses a tree broadcast algorithm to achieve an average transmission time that increases only logarithmically with the number of receivers. Recall that in the Shuffle phase of the proposed CDC scheme, every node in a subset \mathcal{S} of $r + 1$ nodes multicasts coded data segments to the rest of r nodes in \mathcal{S} . Therefore, if we implement CDC using `MPI_Bcast`, the actual performance gain (measured by the average duration of data shuffling) over the uncoded scheme would be $\Theta(\frac{r}{\log(r)})$. As a result, adopting CDC in current systems can still substantially reduce the communication load, in spite of lacking network-layer multicast capability.

More generally, we can consider the cost of multicasting which is defined as follows:

Definition 4 (Multicasting Cost). For $k \in \{1, \dots, K\}$, we define the k -multicasting cost, denoted by $\rho(k)$, as the average transmission time of multicasting one bit to k nodes. \diamond

For example, in distributed computing platforms that exchange information via wireless links, which include wireless data centers [25], [26] and distributed mobile computing platforms [27], [28], multicasting one bit to k nodes is roughly as fast as unicasting one bit to a single node, i.e., $\rho(k) \approx \rho(1) = \Theta(1)$. However, this assumption is too optimistic for wired networks. For instance, using the MPI broadcast whose performance is plotted in Fig. 5, the k -multicasting cost $\rho(k) = \Theta(\log(k))$.

With this definition, one interesting problem is to minimize the communication load weighted by the multicasting cost ($\rho(r)L_{\text{coded}}(r)$ achieved by CDC and $\rho(1)L_{\text{uncoded}}(r)$ achieved by the uncoded scheme), or roughly speaking, the time spent for data shuffling.

B. Joint Storage and Computation Optimization

In the problem formulation, we assumed that we can design the placement of the input files such that each node k , $k \in \{1, \dots, K\}$ locally stores the files in \mathcal{M}_k that it needs to process in the following Map phase. However, in practical file storage systems (e.g., GFS [29] and HDFS [30]), data blocks are often stored without prior knowledge about the computations that will be performed on them.

We claim that without the capability of designing the data placement based on the computation job, one can still achieve a communication load only slightly higher than what is achieved by CDC that assumes full control of the data placement. First, we observe that the data redundancy already exists in practical file storage systems (e.g., GFS and HDFS by default place replicas of each data block on 3 distributed nodes). Using such existing repetitive data placement, the coded shuffling scheme of CDC alone (without requiring storing input files based on the Map phase design in Section IV-A) can take advantage of coding opportunities to substantially reduce the communication load.

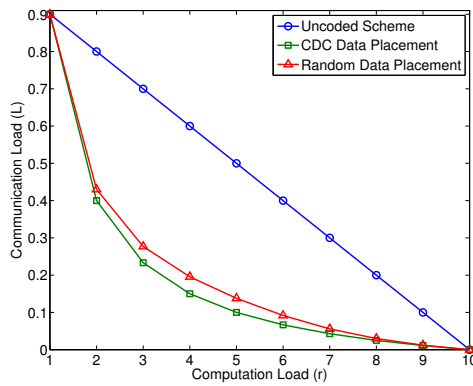


Fig. 6: Comparison of the average communication load by placing and mapping every input file randomly at r out of $K = 10$ nodes with the communication load achieved by placing the files specified by the CDC scheme. Here we compute $Q = 10$ output functions from 2520 input files using $K = 10$ distributed computing nodes.

We plot in Fig. 6 the average communication load achieved by a coded shuffling scheme similar to the one presented in Section IV-B (with the modification that the each node zero-pads its associated data segments to the length of the longest one before coding), when each input file is placed and mapped at r out of K nodes chosen

uniformly at random, and compare it with the communication load achieved by CDC where the input files are placed based on the Map phase design in Section IV-A. As demonstrated in Fig. 6, without requiring the files to be placed as exactly described by the CDC scheme, one can still exploit the data redundancy to achieve a communication load that is superlinear with respect to the computation load. Therefore, the coded data shuffling scheme of CDC can be applied to computation jobs on general data storage systems. In the context of caching, this behavior has been reported in [17].

C. Synchronicity

We assumed in the problem formulation that the Shuffle phase of the distributed computing framework does not start until all the nodes have mapped all the input files they are responsible for. However, the mappings of the input files and the data shuffling process can often be parallelized. Particularly in the proposed CDC scheme, knowing that the intermediate values required to decode the coded messages will eventually be computed by the receiver node, the transmitter node can generate and multicast the coded messages once it has computed the intermediate values needed for coding. More specifically, in the illustrative example depicted in Fig. 4, Node 1 can multicast the coded message $v_{1,2} \oplus v_{2,1}$ once it has computed $g_2(w_2)$ and $g_1(w_1)$. Also Node 1 can compute $g_4(w_4)$ which contains the intermediate value $v_{4,1}$ useful for decoding, *after* it receives the coded messages $v_{4,1} \oplus v_{1,4}$ from Node 2. As a result, it is completely valid (even preferred) for the Map phase and the Shuffle phase of the proposed CDC scheme to overlap with each other.

VIII. CONCLUDING REMARKS AND FUTURE DIRECTIONS

We introduced a scalable distributed computing framework motivated by MapReduce, which is suited for arbitrary types of output functions. We formulated and exactly characterized an information-theoretic tradeoff between computation load and communication load within this framework. In particular, we proposed Coded Distributed Computing (CDC), a coded scheme that reduces the communication load by a factor that can grow with the network size, illustrating the role of coding in speeding up distributed computing jobs. Furthermore, we proved a tight information-theoretic lower bound on the minimum communication load, using any data shuffling scheme, which exactly matches the communication load achieved by CDC. This result reveals a fundamental relationship between computation and communication in distributed computing—the two are inverse-linearly proportional to each other.

Next we highlight two future directions of this work. Firstly, in this paper we assumed arbitrary multicasting networks. It would be interesting to explore the impact of coding in saving the bandwidth requirement for particular network topologies. Examples of such topologies include the fat-tree topology for large-scale computer clusters [31] and the star topology commonly seen in WiFi networks and the emerging mobile edge computing paradigm. Secondly, we can consider a general cascaded distributed computing framework that is comprised of D rounds of MapReduce computations. Given a computation load r_d for the Map phase of the d th round, $d \in \{1, \dots, D\}$, it would be interesting to characterize the minimum overall communication load across all D data shuffling processes.

REFERENCES

- [1] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, “Coded MapReduce,” *53rd Annual Allerton Conference on Communication, Control, and Computing*, Sept. 2015.
- [2] —, “Fundamental tradeoff between computation and communication in distributed computing,” *IEEE International Symposium on Information Theory*, July 2016.
- [3] J. Dean and S. Ghemawat, “MapReduce: Simplified data processing on large clusters,” *Sixth USENIX Symposium on Operating System Design and Implementation*, Dec. 2004.
- [4] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, “Spark: cluster computing with working sets,” in *Proceedings of the 2nd USENIX HotCloud*, vol. 10, June 2010, p. 10.
- [5] Y. Guo, J. Rao, and X. Zhou, “iShuffle: Improving Hadoop performance with shuffle-on-write,” in *Proceedings of the 10th International Conference on Autonomic Computing*, June 2013, pp. 107–117.
- [6] F. Ahmad, S. T. Chakradhar, A. Raghunathan, and T. Vijaykumar, “Tarazu: optimizing MapReduce on heterogeneous clusters,” in *ACM SIGARCH Computer Architecture News*, vol. 40, no. 1, Mar. 2012, pp. 61–74.
- [7] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, “Managing data transfers in computer clusters with orchestra,” *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 98–109, Aug. 2011.
- [8] A. C.-C. Yao, “Some complexity questions related to distributive computing (preliminary report),” in *Proceedings of the eleventh annual ACM symposium on Theory of computing*, Apr. 1979, pp. 209–213.
- [9] J. Körner and K. Marton, “How to encode the modulo-two sum of binary sources,” *IEEE Transactions on Information Theory*, vol. 25, no. 2, pp. 219–221, Mar. 1979.
- [10] A. Orlitsky and A. El Gamal, “Average and randomized communication complexity,” *IEEE Transactions on Information Theory*, vol. 36, no. 1, pp. 3–16, Jan. 1990.
- [11] K. Becker and U. Wille, “Communication complexity of group key distribution,” in *Proceedings of the 5th ACM conference on Computer and communications security*, Nov. 1998, pp. 1–6.
- [12] E. Kushilevitz and N. Nisan, *Communication Complexity*. Cambridge University Press, 2006.
- [13] A. Orlitsky and J. Roche, “Coding for computing,” *IEEE Transactions on Information Theory*, vol. 47, no. 3, pp. 903–917, Mar. 2001.
- [14] B. Nazer and M. Gastpar, “Computation over multiple-access channels,” *IEEE Transactions on Information Theory*, vol. 53, no. 10, pp. 3498–3516, Oct. 2007.
- [15] A. Ramamoorthy and M. Langberg, “Communicating the sum of sources over a network,” *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 4, pp. 655–665, Apr. 2013.
- [16] M. A. Maddah-Ali and U. Niesen, “Fundamental limits of caching,” *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2856–2867, Mar. 2014.
- [17] —, “Decentralized coded caching attains order-optimal memory-rate tradeoff,” *IEEE/ACM Transactions on Networking*, Apr. 2014.
- [18] M. Ji, G. Caire, and A. F. Molisch, “Fundamental limits of caching in wireless D2D networks,” *IEEE Transactions on Information Theory*, vol. 62, no. 2, pp. 849–869, Feb. 2016.
- [19] N. Karamchandani, U. Niesen, M. A. Maddah-Ali, and S. Diggavi, “Hierarchical coded caching,” *IEEE International Symposium on Information Theory*, pp. 2142–2146, June 2014.
- [20] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, “Speeding up distributed machine learning using codes,” *e-print arXiv:1512.02673*, 2015.
- [21] A. Rajaraman and J. D. Ullman, *Mining of massive datasets*. Cambridge University Press, 2011.
- [22] Y. Birk and T. Kol, “Coding on demand by an informed source (ISCOD) for efficient broadcast of different supplemental data to caching clients,” *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2825–2830, June 2006.
- [23] Z. Bar-Yossef, Y. Birk, T. Jayram, and T. Kol, “Index coding with side information,” *IEEE Transactions on Information Theory*, vol. 57, no. 3, pp. 1479–1494, Mar. 2011.
- [24] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, “Network information flow,” *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.
- [25] D. Halperin, S. Kandula, J. Padhye, P. Bahl, and D. Wetherall, “Augmenting data center networks with multi-gigabit wireless links,” in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, Aug. 2011, pp. 38–49.

- [26] Y. Zhu, X. Zhou, Z. Zhang, L. Zhou, A. Vahdat, B. Y. Zhao, and H. Zheng, “Cutting the cord: a robust wireless facilities network for data centers,” in *Proceedings of ACM 20th annual international conference on Mobile computing and networking*, Sept. 2014, pp. 581–592.
- [27] G. Huerta-Canepa and D. Lee, “A virtual cloud computing provider for mobile devices,” in *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*. ACM, 2010, p. 6.
- [28] H. Ba, W. Heinzelman, C.-A. Janssen, and J. Shi, “Mobile computing—A green computing resource,” in *IEEE Wireless Communications and Networking Conference (WCNC)*, Apr. 2013, pp. 4451–4456.
- [29] S. Ghemawat, H. Gobioff, and S.-T. Leung, “The Google file system,” in *ACM SIGOPS operating systems review*, vol. 37, no. 5, Dec. 2003, pp. 29–43.
- [30] “Apache Hadoop Distributed File System,” https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html.
- [31] M. Al-Fares, A. Loukissas, and A. Vahdat, “A scalable, commodity data center network architecture,” *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 63–74, Oct. 2008.